This is the authors' accepted manuscript, the article has been accepted for publication in the Journal of the Audio Engineering Society: Anna Xambó, Gerard Roma, Pratik Shah, Takahiko Tsuchiya, Jason Freeman, and Brian Magerko (2018). Turn-taking and Online Chatting in Co-located and Remote Collaborative Music Live Coding. *Journal of the Audio Engineering Society*, 66(4), 253256.
Please refer to the published version here: `http://www.aes.org/e-lib/browse.cfm?elib=19391`.

Turn-taking and Online Chatting in Co-located and Remote Collaborative Music Live Coding

ANNA XAMBÓ
(a.xambo@qmul.ac.uk)

Queen Mary University of London, London, UK
Georgia Institute of Technology, Atlanta, USA

GERARD ROMA
(g.roma@hud.ac.uk)

University of Huddersfield, Huddersfield, UK
Georgia Institute of Technology, Atlanta, USA

PRATIK SHAH, TAKAHIKO TSUCHIYA, JASON FREEMAN, BRIAN MAGERKO
(prats110892@gmail.com, takahiko@gatech.edu, jason.freeman@gatech.edu, magerko@gatech.edu)

Georgia Institute of Technology, Atlanta, USA

Collaborative music live coding (CMLC) approaches the music improvisation practice of live coding in collaboration. In network music, co-located and remote interactions are possible, and communication is typically supported by the use of a chat window. However, paying attention to simultaneous multi-user actions, such as chat texts and code, can be demanding to follow. In this paper, we explore co-located and remote CMLC using the live coding environment and pedagogical tool EarSketch. In particular, we examine the mechanism of turn-taking and the use of a small set of semantic hashtags in online chatting by using an autoethnographic approach of duo and trio live coding. This approach is inspired by (1) the practice of pair programming, a team-based strategy to efficiently solving computational problems; (2) the language used in short messaging service (SMS) texting and social media. The results from an online survey with six practitioners in live coding and collaboration complements the autoethnographic findings and point to education as the most suitable domain for this approach to CMLC. We conclude discussing the challenges and opportunities of turn-taking and the use of semantic hashtags focusing on educational settings.

0 INTRODUCTION

Live coding and collaboration is a promising approach to the established discipline of computer-supported cooperative work (CSCW) in the musical domain. Collaborative music live coding (CMLC) brings collaboration into live coding, a well-known musical improvisation practice in computer music. Using chat mechanisms has been found to support communication between the either co-located or remote live coders in CMLC [1, 2, 3, 4]. A similar collaborative live coding practice that organizes the group workflow in a turn-taking fashion between the roles of a driver and a navigator is pair programming [5], which is used in industry and computer science (CS) education.

Inspired by both CMLC and pair programming, in this paper we explore the promises and challenges of supporting communication features for turn-taking pair programming in EarSketch [6], a live coding environment and educational tool. We particularly explore the use of semantic hashtags in a chat window and a shared code editor, simi-

lar to Google Docs collaborative editing, in both co-located and remote pair programming, with a group of two and three users respectively. With a few exceptions [7], editing a shared single script is a novel approach to CMLC, where we typically find examples of live coders editing their own individual scripts (e.g., JITlib [8]) or individual shared editors where each live coder can edit others' editors (e.g., Gabber [9]).

Our main research question is: *How do music live coders collaborate, and what design paradigms for live coding tools can effectively support these collaboration modes?* In particular, we aim at examining whether and how turn-taking supports collaborative live coding. We are interested in understanding: (1) duo vs trio conditions: what can be the tasks undertaken by the different collaborators based on the turn-taking and pair programming model of driver vs navigator(s) and whether there are any differences between the configurations of one driver vs one navigator and one driver vs two navigators; (2) using a chat window: whether a chat window is an effective mechanism for communication and, if so, how it is used; (3) using semantic hashtags: whether using a small set of semantic hashtags that enforce turn-taking in the chat window is helpful in a collaborative session; (4) co-located vs remote conditions: what are the main differences between the co-located and remote conditions. We adopted an autoethnographic approach already used for understanding a live coding environment [10] because it can help to gain a first thorough insight from a live coder perspective.

This study builds on our prior research on co-located CMLC using EarSketch [11]. This study contributes with a comparison between co-located and remote conditions with the same duo and trio of live coders. The results of an online survey with six practitioners in live coding and collaboration complement our findings, which indicated the following: (1) remote turn-taking and online chatting is a promising approach in CS education that can be used in both onsite and online classes; (2) turn-taking in performance can easily become too predictable, both in co-located and remote settings, and thus should be combined (if used it at all) with other collaboration strategies; (3) the use of own invented hashtags (as opposed to provided hashtags) in online chatting is a promising approach to peer communication, especially in education.

1 RELATED WORK

This section presents relevant work within the context of computer-supported cooperative work (CSCW) and computer music live coding (CMLC), where we highlight: first, relevant performance and education live coding environments, communication tools in CSCW and CMLC, and our related prior work; second, research on turn-taking and pair programming; third, language used in SMS and social media.

1.1 CSCW and CMLC

The field of CSCW looks at how group activities can be computationally supported. Barbosa [12] proposes a classification of computer-supported collaborative music (CSCM) systems based on the dimensions of space and time, inspired by a classification space for CSCW: location of players (*co-located* vs *remote*), and performance synchronicity (*synchronous* vs *asynchronous*). *Synchronous/co-located* is exemplified by interactive tabletops; *synchronous/remote* is exemplified by videoconferencing; *asynchronous/co-located* is exemplified by a post-it interface; and *asynchronous/remote* is exemplified by a version control system. This research focuses on synchronous interaction in CSCM, both co-located and remote.

1.1.1 Performance Environments for CMLC

Small group collaboration using individual screens and working as a network with a shared clock is reported in [13]. For example, Gibber implements network time synchronization that is controlled by a master server where each client needs to adjust any differences [9]. A common collaborative live coding configuration is working in pairs [14]. Collaborative live coding in large groups, under democratic rules, is investigated by the Republic system [15]. The recent advent of the Web Audio API,[1] in JavaScript, has enabled a number of projects to explore the possibilities of mobile orchestras and participatory audience with their mobile devices using their browser and connected to a network [16]. With a few exceptions, such as the UrMus live coding language for mobile phones used extensively by the Michigan Mobile Phone Ensemble [17], there is little research on live coding on mobile devices, particularly participatory live coding, which seems an interesting direction to computationally support collaboration among large groups.

1.1.2 Learning Environments for CMLC

Scratch[2] is a blocks-based programming language, which can be used to create a variety of multimedia applications, most often games. Scratch has been used in music-related projects, for example, for teaching computational thinking through music in the classroom [18]. Live coding comes into play when using infinite loops and algorithmic composition (cf. [19]). Another example is Sonic Pi,[3] an open source live coding environment for teaching computing concepts through music, which runs on the tiny computer Raspberry Pi [1]. To our knowledge, platforms for live coding in learning environments have been researched and designed focusing on individual use, as opposed to collaborative use as a key component in education.

There is a significant amount of research on the pedagogical benefits of live coding in education, as evidenced

[1] http://www.w3.org/TR/webaudio
[2] http://scratch.mit.edu
[3] http://sonic-pi.net

in the special issue on "Live Coding for Music Education," published in the *Journal of Music, Technology & Education* [20]. This issue included an analysis of the challenges and opportunities of teaching live coding in the classroom using EarSketch [21]. Our focus on CMLC in education is a follow-up of this work.

1.1.3 Communication Tools in CSCW and CMLC

There exist a number of environments that support real-time collaboration for text editing (e.g., SubEthaEdit,[4] ShareLatex,[5] Overleaf);[6] learning (e.g., Virtual Math [22]); and coding (e.g., CodeCircle [23]). The most common characteristics for supporting real-time collaboration, in alignment with CSCW design recommendations, include: (1) shareable links; (2) users' online presence; (3) color coding users; (4) levels of access permissions; (5) communication tools (e.g., chat, commenting); (6) visual feedback of real-time multiple-interaction; (7) version history; (8) error handling as a shared experience.

Facilitating networked collaboration and communication is reported as essential for CMLC [3]. As a follow-up of Section 1.1.1, music live coding environments that support communication tools include Overtone [1], Gibber [24], Betablocker [25], the Republic quark [15] in SuperCollider, and Fluxus [25]. In Overtone, there is a text chat panel that lets the members of a session send messages separated from the code. In Gibber, the system Gabber allows users to meet in a chat room where shared editing of code documents is possible [9]. Editing is in sync, but execution is independent for each user. Other environments that support co-located and remote text-based collaborative music improvisation and include a chat window are LOLC [2], urMus [3], and Lich.js [4]. A study on a CSCW-based music system reported that the chat was used for exchanging descriptions of the users' activities [26].

Our paper is based on the use of the communication tools traditionally used in CSCW for supporting awareness and communication among team members, such as a chat, a buddy list, and a code editor as a shared space. Turn-taking in EarSketch is a first step towards supporting real-time collaboration using these tools. How these tools might be used effectively in CMLC is an open question that we investigate in this study.

1.1.4 Our Prior Studies

This study is a follow-up of the studies reported in [14, 27, 11]. Co-located human-human interaction in education has been explored in [14], where we proposed three approaches to real-time collaboration: (1) a shared script accessed from a single terminal in turns (*live pair programming*); (2) a shared script accessed from multiple individual terminals at any time (*multiple live coding*); (3) a shared script accessed from two individual terminals at any time (*pair live coding*). Co-located CMLC focusing on turn-taking is investigated in [11], where we explored a merged version of the above three approaches to CMLC. Each user had an individual terminal (like in *pair* or *multiple live coding*), the roles were strictly divided into driver and navigator (like in *live pair programming*), with the addition of a second navigator for trio live coding (like in *multiple live coding*). In this study we expand the turn-taking work by also looking at remote CMLC.

1.2 Turn-taking and Pair Programming in CSCW

The organization of turn-taking has been studied in conversation analysis since late 1960s and early 1970s [28]. Supporting turn-taking has been a topic of research of CSCW and groupware technology since early 1990s [29]. Interaction analysis has included both verbal and non-verbal communication when analyzing technology-enhanced collaborative settings [30], where both *talk-driven interaction* (e.g., 'turns at talk') and *instrumental interaction* (e.g., 'turns with bodies' and 'turns with artifacts', such as two kids taking turns with a mouse when playing a game) are examined.

Pair programming [5] is an established practice in CS where two programmers, a driver and a navigator, collaborate on the same computational problem (e.g., designing, coding or testing) and after a certain amount of time (e.g., 10 minutes) they switch roles. Whilst the driver is in charge of writing the code, the navigator is responsible of both the long-run thinking, in particular the nitty-gritty details and potential errors, such as syntax errors. Turn-taking between a small number of users is described in a seminal paper [31] on designing systems for collaborative musical experiences as a useful protocol for assessing new musical instruments, notably in [32, 33]. This study is influenced by pair programming practices, but supporting larger small groups than pairs in a turn-taking fashion of a driver and a few navigators, and applied to the domain of music computing, which is novel in the literature.

1.3 Language in SMS and Social Media

The literature indicates that using short messaging service (SMS) texting and social media hashtags do not interfere with formal writing [34, 35]. Features of SMS messaging include the use of short utterances, abbreviations, emoticons, and other symbols [36]. Syntactical and lexical reductions are applied in SMS texting to reduce effort, time, and space [37]. These practices have extended to chats, instance messaging, tweets, and Internet-related communication in general.

Our approach to SMS and social media hashtags is to take advantage of their established use among students but applied to facilitate the communication between live coders during a CMLC session. Here the use of the SMS texts and social media hashtags benefits an informal learning environment where efficient communication is in the fore. Supporting an informal language style, commonly used between students, can increase the intention to persist in programming, which is a well-known challenge in CS education [6].

[4] `http://codingmonkeys.de/subethaedit`
[5] `http://www.sharelatex.com`
[6] `http://www.overleaf.com`

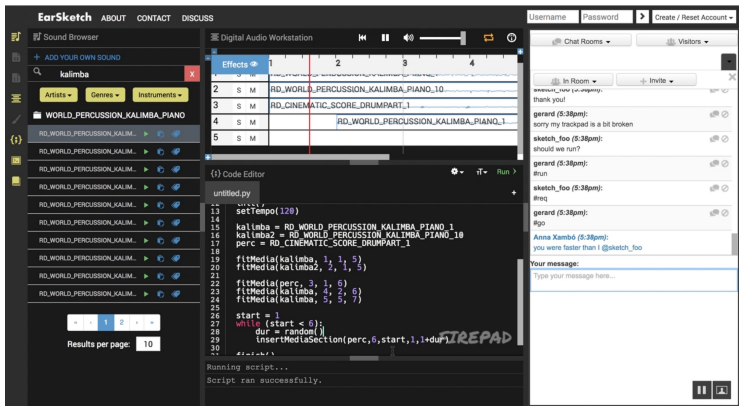


Fig. 1. Screenshot of the proof-of-concept prototype of EarSketch with a collaborative code editor and a chat window.

2 CMLC & EARSKETCH

EarSketch [6] is a free online tool and curriculum designed for learning to code Python or JavaScript languages by making music with audio samples, beats, and effects. It is inspired by a digital audio workstation (DAW) interface and supports both composition and live coding features [14]. Next we detail the collaboration capabilities of EarSketch and the prototype version used in this study.

2.1 EarSketch and Collaboration

Collaboration in EarSketch has been progressively implemented. The first phase introduced the capability of sharing scripts through links or users and sharing the audio on SoundCloud. Another feature added was to automatically retain the metadata of the original script, such as the title and author name. Using the workflow of importing the code before allowing users to edit ensures that the authorship is attributed when the users share or publish the script. This study is designed as an early prototype of the planned second and third phases of collaboration in EarSketch. The second phase will allow multiple users to work on the same script, permitting the possibility of real-time collaborative script editing. The final phase plans to introduce features that support peer-to-peer communication using a shared chat window.

2.2 Proof of Concept

To explore our research question, we developed a prototype version of EarSketch (see Figure 1) that added basic collaborative simultaneous editing features and a chat window. The aim was at informing next iterations of the web-based software. We used Firepad[7] and Firechat[8] to include simultaneous code editing and online chatting, respectively. Both applications are built on Firebase.[9]

[7] http://firepad.io
[8] http://firechat.firebaseapp.com
[9] http://firebase.google.com

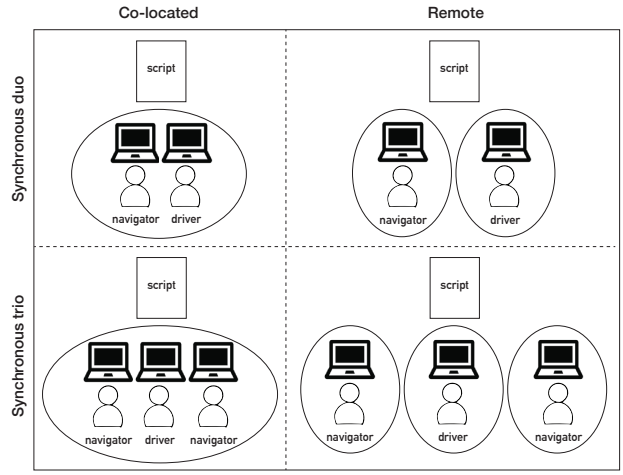


Fig. 2. Overview of the four use cases of our autoethnographic study.

3 THE STUDY

In this section, we present the design of the autoethnographic study and the survey to live coder practitioners about turn-taking and online chatting in CMLC.

3.1 Autoethnographic Study

For an overview of the four use cases of the autoethnographic study, which are based on the number of performers (two vs three) and presence type (co-located vs remote), see Figure 2. In this section, we detail the task, protocol, and research methods used for the four situations.

3.1.1 Task and Protocol

The task consisted in creating four 15-minute live coding sessions, two co-located and two remote, with a duo and a trio for each location (see Figure 2). For each live coding session, a group of live coders was asked to work on a shared blank Python script of EarSketch using their own individual laptops. The group was expected to write code by turn-taking and to communicate between the team members (e.g., taking decisions) via the chat window. Each participant was expected to act as driver at least once in each

session. The group was encouraged to communicate only using an embedded chat window with a mindset of a performance setting. The first person to start writing code was suggested to be chosen by agreement of the group. In the co-located condition, the assumptions were that performers are located in the same location and that one computer connected to a PA system renders locally the musical performance. In the remote condition, the assumptions were that performers are located at different locations and that each computer renders locally the musical performance.

For facilitating turn-taking in the chat window, the use of the following semantic hashtags was suggested to the group:

- *#req* for requesting a turn;
- *#go* for giving the turn;
- *#run* for running the code.

In the co-located condition, the latter hashtag meant that the driver would execute the code to be sent to the PA system. In the remote condition, both driver and navigator(s) would execute the code to be sent to their own headphones. Direct messages were encouraged using the username preceded with an at sign (e.g., *@username*).

3.1.2 Participants

The three participants (L1–L3) were expert musicians with extensive experience in live coding. One of them, who performed as part of the trio, was an expert in EarSketch. The same co-located duo and trio groups participated in the remote experience.

3.1.3 Data Collection and Analysis

Both scripts and comments in the chat window were stored for qualitative data analysis. The session was video recorded using the screencast software SnagIt.[10] In the co-located condition, the screencast software captured the screen from the computer that was connected to the loudspeakers while the group was working with EarSketch. In the remote condition, the screencast video was captured from one of the computers. We were inspired by an autoethnography perspective of reflecting on our own live coding practice already used in live coding [10]. In the co-located condition, we had conversations about the experience afterwards, which were combined with watching the videos and reviewing the chat texts. Whilst in the remote condition, we had a broader perspective and answered an online questionnaire, which is explained next.

3.1.4 Survey

For the remote condition, we asked the live coders to fill in a post-questionnaire with five open questions (see Appendix I). They were framed within performance and education; the overall experience (co-located vs remote); the turn-taking approach to live coding; and the suitability of the number of performers.

3.2 Live Coding Community Study

We also conducted an online survey addressed to live coder practitioners to obtain feedback about their practices and thoughts about collaborative music live coding, which the intention to help framing the findings from the autoethnographic study.

3.2.1 Task and Protocol

Participants were asked to (1) watch a 20-minute video[11] of CMLC (which was a video extract of the remote live coding sessions from the autoethnographic study) and (2) fill in an online survey. The overall session was designed to take 35 minutes or less to complete. The video was shown as a proof of concept of turn-taking and online chatting in collaborative music live coding. The video showed two approaches to CMLC using EarSketch: (1) trio live coding; (2) duo live coding. The aim of watching this video was to elicit ideas around CMLC that were going to be discussed in the survey.

3.2.2 Survey

We were interested in surveying musicians who are practitioners of live coding to gain insight into their practices and expectations when collaborating with other live coders (see Appendix II). In particular, we were trying to better understand how live coders communicate while coding, the roles they take, and the type of graphical user interfaces (GUIs) or chat interactions they would prefer (e.g., emoticons, hashtags, notifications on the screen, side comment on the code they exchange, multimodal notifications such as buzzers on the body or sounds). The survey combined open-ended questions with 5-point Likert item questions (from strongly disagree to strongly agree and 0 as N/A).

3.2.3 Participants

We sent an invitation to fill in the survey to some live coding practitioners. We obtained responses from 6 male live coder practitioners (P1–P6), with an age range of 25–34 years old (1 participant) and 35–44 years old (5 participants). One had 7–9 years of experience with music as a practitioner whilst 5 had more than 10 years of experience. Their experience with live coding was diverse, ranging from less than 2 years (1 participant), 4–6 years (2 participants), and more than 10 years (3 participants). Similarly, their experience with collaborative music live coding ranged from less than 2 years (2 participants), 2–3 years (1 participant), 4–6 years (1 participant), and more than 10 years (2 participants). The tools/technologies used for live coding were varied, where participants tend to use more than one tool, including SuperCollider (4 participants), TidalCycles (2 participants), LOLC (1 participant), ChucK (1 participant), PureData (1 participant), Python (1 participant), Gibberwocky (1 participant), and own-built environments (1 participant). The tools used to support

[10] http://www.techsmith.com/screen-capture.html

[11] Video of remote duo vs trio live coding: http://vimeo.com/242667005

Table 1. Overview of findings from the autoethnographic study

	Co-located	Remote
Duo	(1) Dynamic change of roles, participation, predictable linear interactions, non-specialist roles (request/grant control), little attention to the others' actions.	(1) Smooth role negotiation, non-specialist roles (request/grant control).
Duo	(2) Peer learning, suggestions, planning, opinions, reflections, coordination, communication between the team.	(2) Extensive use of chat, salutations, peer learning, suggestions, planning, opinions, reflections, coordination, collaboration, dialogue about what each other is doing.
Duo	(3) *#req*, *#go*, *#run* used; *@username* not used.	(3) *#req*, *#go*, *#run* used; *@username* not used.
Duo	(4) Direct communication (e.g., verbal), physical awareness.	(4) Lack of physical awareness of others, lack of visual cues.
Trio	(1) Organic change of roles, specialist roles, attention to the others' actions.	(1) Non-specialist roles (request/grant control), competitiveness between navigators, inefficient change of roles.
Trio	(2) Peer learning, suggestions, planning, opinions, reflections, coordination, communication between the team.	(2) Extensive use of chat, salutations, peer learning, suggestions, planning, opinions, reflections, coordination, collaboration, communication with the driver.
Trio	(3) *#req*, *#go*, *#run* used; *@username* used.	(3) *#req*, *#go*, *#run* used; *@username* used.
Trio	(4) Direct communication (e.g., verbal), physical awareness.	(4) Lack of physical awareness of others, lack of visual cues.

their collaborative live coding sessions include SuperCollider (3 participants), LOLC (1 participant), APICultor (1 participant), and code/music sharing via Slack, GitHub and Lurk.org (1participant).

4 FINDINGS

In this section, we report and compare the four use cases of trio and duo live coding in co-located and remote settings from an autoethnographic perspective. Then we summarize the results from the online survey conducted with live coder practitioners.

4.1 Autoethnographic Study

4.1.1 Overview

We did two sessions of co-located CMLC in Atlanta (GA, USA) with, a trio and a duo, respectively.[12] One laptop was connected to two loudspeakers and a live coder was responsible to run the script when one of the live coders asked for it. The other one or two live coders were wearing headphones that allowed them to preview sounds or scripts independently, if necessary. In the remote condition,[13] the audio was not broadcasted via loudspeakers as in the co-located condition. Live coders were wearing headphones instead. The rationale was that participants would have the same "performance-with-no-latency" experience (as opposed to only one listening to the PA and the audio feeding back to the performers with latency). As shown in Figure 3, the trio members were located in Atlanta (GA, USA), London (UK), and Huddersfield (UK), whilst the duo members were located in London and Huddersfield. The four sessions lasted around 20 minutes each.

Table 1 overviews the main findings from the autoethnographic study, which is structured according to our research interests presented in Section 0 about (1) duo vs trio conditions; (2) using a chat window; (3) using semantic hashtags; (4) co-located vs remote conditions. The table matches the study design with summaries of the performance strategies used in each condition.

4.1.2 Co-located Trio Live Coding

Two navigators and one driver allowed us to have more specialist roles. One task of the navigator was to preview sounds with their headphones and suggest audio samples' names that could suit well. This task was done autonomously or requested by another live coder. If there was an error when executing the code from the main terminal, previewing the error from another terminal was also used to solve the problem. As the piece included algorithmic elements, every time that we pressed the *run* button there was a change and thus executing the code multiple times produced an interesting variety of musical results. We ended the piece with a combination of quick changes and executions of the code that resulted in a compelling glitch style.

We constantly used the chat window to explain unclear code snippets (e.g., *"master track would be MASTER_TRACK instead of track number"* (L3)), share plans of individual actions (e.g., *"I want to make a longer beat string algorithmically"* (L3)), or suggest actions that others could do (e.g., *"maybe mute the tracks one by one?"* (L3)). The chat window was also used to plan parts of the piece together (e.g., *"btw we can start doing an ending?"* (L1)), ask for opinion or help (e.g., *"are any of the groove sounds good here? e.g. HOP_DUSTYGROOVEPART_001"* (L1)), and or comment about the musical results (e.g., *"this is nice"* (L3)). Having three on board, allowed us to tell others when we were done or tired of driving the session (e.g., *"and now ready for someone else with fresh ideas..."* (L3)). The three suggested tags were used for requesting, granting, and asking the live coder of the master terminal to execute code. However, direct or group messages were generally assumed without making them explicit. For example, *@username* was occasionally used and there were

[12] Video of co-located duo vs trio live coding: http://vimeo.com/212639022

[13] Video of remote duo vs trio live coding: http://vimeo.com/242667005

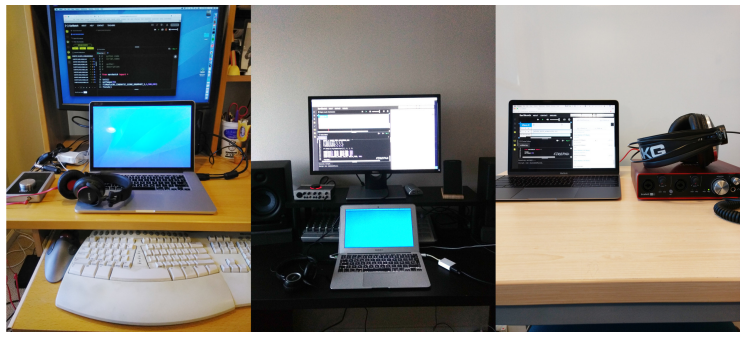


Fig. 3. The setup of the remote trio live coding. From left to right: Atlanta (GA, USA), Huddersfield (UK), and London (UK).

instead other messages targeting individuals. We requested an equal frequency of turns during the session. The expert live coder of EarSketch helped the team when using other functions than the common `fitMedia()` or `makeBeat()`. For example, the expert live coder explained the `makeBeatSlice()` function via the chat window (e.g., *"each successive number is an index to a different timestamp in the sound"* (L3)). This saved us time of looking at the EarSketch curriculum during the performance session and was an instance of situated peer learning [38].

4.1.3 Co-located Duo Live Coding

If compared to the trio live coding, we also equally requested turns during the session, but the pace was faster. From a driver perspective, it was more participative, but it also felt more linear, like in a ping pong game of back and forth with the ball, thus *@username* was not needed. The session felt more predictable. In particular, the roles were less specialized, focusing on alternating between driving and navigating. Often times, the navigator was planning her or his next move as a driver and paying less attention to the driver's actions because there was the impression of time constraints. The chat window was used similar to how we used it in the trio live coding session: as a tool for communication to discuss about the code and the musical output.

4.1.4 Remote Trio Live Coding

Here the navigators had less specialised roles and were more focused on requesting and granting control. For example, there were no instances of navigators suggesting sounds to the driver. There were instances instead of asking the driver to change and improve their own code written previously, e.g., *"can you change in line 19 kalimba to kalimba 2?"* (L1) or *"can you change line 33 from 0.05 to 0.25? It's mostly selecting a silent portion of the audio file."* (L3). Turns were taken equally but the navigators needed to be alert in order to take the opportunity to drive, otherwise it was easy that the two fastest live coders were alternating the lead. Navigators took any opportunity to request control from the driver (*"I can add a sound while you chat, #req"* (L2)) but sometimes had to wait considerably before their request was granted (e.g., when fixing an unknown error). Random and algorithmic effects were explored (e.g., an array with a list of sounds that could be picked randomly), which took effect every time the run button was pressed.

The chat window was extensively used to communicate salutations (e.g., *"hi"* (L3)), remember the meaning of the hashtags (*"do you mean #run?"* (L1)); requesting and giving control with words in support of the hashtags (*"I want to give you control now"* (L3)); asking changes in the code on their behalf (see above); commenting about how to change the code in case of an error or doubt (*"select from a python list using random"* (L3), *"you are using it wrong, `selectRandomFile()` wants a folder constant, not a bunch of file names"* (L3)); commenting about the musical results or song structure (e.g., *"those drums are lonely in measure 6"* (L3), *"let's do a solo and then end"* (L2)); and even commenting about how difficult was to get a turn or lack of time to complete an idea (e.g., *"but I never finished my idea"* (L3)). There were also instances of expressing the need to solve an error (e.g., *"I'm getting overlapping errors"* (L2), *"it's not playing my track"* (L2)) or solving errors in collaboration (e.g., discussing an alternative random function to `rand()` within EarSketch, like `randint()`). Direct messages were used with and without *@username*, as it was assumed that both navigators were mostly talking to the driver. Given the lack of visual cues of seeing the other musicians not particularly reinforced with the proof of concept either (e.g., lack of information about idle users), it was hard to tell what was happening beyond the code editor and the chat window.

4.1.5 Remote Duo Live Coding

Remote duo live coding was similar to the co-located experience (e.g., non-specialist roles, *@username* was not used in the chat window), but with the lack of the visual cue. For example, it was difficult to tell whether the other musician was active or idle: there was a situation where the driver of that moment got disconnected from the Internet, but the navigator did not notice it from the visual feedback of the proof-of-concept interface: *"it seems I was disconnected, back in."* (L2). There were more explanations of the tasks that the other was doing: *"listening to some samples"* (L2). In the duo, it was easier to negotiate the turn than in the trio (e.g., *"let me fix it after your turn"* (L1)). In both duo and trio, there were instances of discussions or suggestions about parameters and functions in the code

(e.g., *"what is the value of* `DELAY_TIME?` *"* (L2) or *"you can create a loop"* (L1)).

4.1.6 Our reflection

The two members who experienced both duo and trio live coding preferred to work as a trio live coding. Partly the fact that one of the live coders is an expert of EarSketch helped. The combination of multiple specialized tasks in co-located interaction makes the improvisation more varied and serendipitous. The live coder needs to be skillful with the programming language, as the musician needs to master the musical instrument, even more if it is only a group of two live coders.

Turn-taking in live coding reminded us of the *cadavre exquis* collaborative game, where the more the merrier for creative discovery, or hip hop concerts with several MCs. However, we agreed that more than 3–4 people would imply fewer opportunities to have control with turn-taking.

In co-located interaction, the preview feature with headphones was useful to the group, which allowed the navigators to preview sounds before adding them to the shared script or solve an error in the code while others were working with different tasks in parallel.

The lack of face-to-face communication in remote collaboration affected the performance. It made more difficult to reinforce certain messages, e.g., be granted control when the driver is engaged. It also made more difficult to imagine the performance as a unified experience, e.g., it was hard to imagine how the music was sounding in the performance venue, mostly because we were equally using individual headphones; or we felt that moments of control as a driver were used more efficiently because navigators were waiting their turn, instead of developing other complementary roles as it happened in the co-located experience.

4.2 Live Coding Community Survey

Here we present the findings from surveying six live coders about CMLC. The live coders expressed their opinions about their thoughts and practice from watching a video of our prototype, but they did not try the prototype themselves.

As shown in Figure 4, participants expressed dislike about working with provided hashtags (e.g. *#req*, *#go*, *#run*) (M=1.5, SD=1.38). The boxplot shows a spread of opinions where the zero value is related to a N/A answer and an outlier above the fourth quartile of a participant who likes to use provided hashtags. Invented hashtags raised slightly more interest (M=1.83, SD=1.47). A participant would use *"comments as #wow and instructions as #pp"* (P1) and another participant would use hashtags for transitions and *"to have some humorous elements as if the musicians are casually collaborating and eventually to enhance audience communication"* (P2). One participant highlighted his preference in avoiding the use of social media technology: *"(I) prefer straightforward listening as a way to collaborate and contribute. I think the music people make sufficient for communication during a performance."* (P6). This contrasts with how we used the provided hashtags as a practical solution to communicate via the online chat in the autoethnographic study.

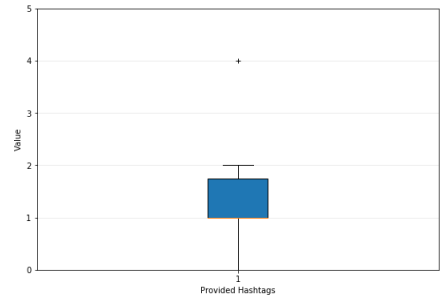


Fig. 4. Boxplot on preference of using provided hashtags for six participants (Mdn = 1.0, M=1.5, SD=1.38).

As illustrated in Figure 5, participants disliked using a turn-taking mechanism mediated by the system during a performance or live coding session (M=1.33, SD=0.82). The boxplot shows that the distribution centers on the value 1, which indicates disagreement, except for an outlier that expressed a neutral opinion. Four participants mentioned turn-taking as a constrained and slow mechanism for live performance and a system better suited for other purposes: *"turn-taking makes the musical change slow and the benefits of it would be rather in something else - pair-programming/educational purposes/exploration/rehearsal/practice rather than actual performance."* (P2). A participant pointed to potential features that can make the turn-taking mechanism more interesting, including *"read-only mode (...), synchronizing viewport (...), and awareness of 'turn'"* (P2). These opinions are in line with the autoethnographic study's findings where we also found turn-taking as a slow mechanism. However, for the co-located trio it worked as an interesting collaborative strategy where different roles emerged.

Participants tended to like the option of multi-editing a shared script (like in Google Docs) during the performance or session (M=3.33, SD=1.37). Participants also tended to like the option of the live coder to be able to edit her or his own script during the performance or session (M=3.0, SD=1.26). In terms of the GUI, participants tended to prefer the use of a color-coding scheme applied to represent the different peers who are contributing to the code/chat during the performance or session (M=3.33, SD=1.75) and the use of a mechanism that shows active vs idle users during the performance or session (M=2.33, SD=1.37). Although our prototype supports multi-editing in a shared script, editing own scripts, and a chat window with a buddy list, more situated information about the users was missed (e.g., color-coding scheme, active vs idle users), which is in line with these results.

An embedded chat was the most popular tool preferred by live coders (3 participants) to communicate between their peers in CMLC, although the other half of the participants were skeptical about online chatting in co-located CMLC. A participant highlighted the preference of combining notifications with messages, to *"see notifications and then read the message"* (P3). Another participant re-

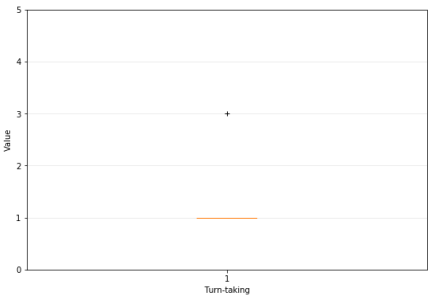


Fig. 5. Boxplot on preference of using turn-taking mechanism for six participants (*Mdn* = 1.0, *M*=1.33, *SD*=0.82).

ported the use of instant messaging apps in remote CMLC (e.g., Skype, Facebook). Two participants expressed their preference for other ways of communicating, either *"voice based"* (P5) or through the activity in the code (*"cursor activity (. . .) when typing in the same buffer"* (P4)). However, a participant raised concern about the difficulty of communicating via laptops: *"Laptops are meant to facilitate communication between a face and a screen. Live coding has to deal with the consequence of this." (P4).* This variation explains the average neutral results from the Likert item question about the preference of using an embedded chat next to the code during the performance or session (*M*=2.66, *SD*=1.86). More research should be done to explore the type of embedded chat that would work best for CMLC situations.

Three of the participants reported their preference to work as a duo live coding. A participant highlighted the benefits of performing with more than two: *"it's more relaxed to keep the music going while changing the own code, and also the possibilities with 3-4 people are greater, since there is more musical input and generates different reactions, and makes the music richer."* (P3). Another participant commented that *"for improvisation in a casual setting - the number does not matter."* (P2). Most of the participants said that they avoid switching roles in their practice (4 participants), two of them in the context of free improvisation (*"I never work with roles, it is an aesthetic choice, I prefer that my collaborator expresses herself in the musical context we create perhaps in the spirit of free improv"* (P6)), whilst one participant commented how the group changes roles fluidly: *"It depends on the music which role whoever does, each member plays."* (P3). Interestingly, from our autoethnographic experience, the roles were paced by the pair programming model (as opposed to musical roles, as discussed here) and we preferred the trio configuration. This points to the multiple approaches to CMLC and how a CMLC system should ideally support this variety.

With the other Likert item questions, there was a diversity of opinions, with tendency to dislike using emoticons (*M*=1.83, *SD*=1.33); using SMS texting (*M*=2, *SD*=1.55); commenting on the side of the code (*M*=2.33, *SD*=1.97); and using multimodal notifications (e.g., buzzers on the body, sounds) (*M*=1.5, *SD*=1.38). Notifications on the screen seemed to be preferred by most of the live coders (*M*=3.33, *SD*=1.97), so that the live coder can concentrate on other tasks. In the autoethnographic study, the provided hashtags were used frequently to communicate via the online chat. A next step might be to link the hashtags to notifications so that the peer communication is more efficient.

5 LESSONS LEARNED

Here we reflect on our practice of turn-taking live coding within the broader picture of co-located and remote CMLC in performance and education. We also discuss the potential of using semantic hashtags connected to the turn-taking mechanism.

5.1 CMLC and Turn-Taking within Education

Turn-taking in CMLC appears to be a useful mechanism for learning how to code. When we had an error, we had comments from our peers on how to fix them. When someone was writing complex code, it was possible to follow from the comments of the other peers. This was particularly reinforced in remote trio live coding, where the two navigators were focused on what the driver was doing. Turn-taking seems suitable for an education setting because it slows down the pace of coding, which can constraint more thoughtful following of peer coding and encourage real-time comments and help.

In duo live coding, the fact that both had a low level of expertise in EarSketch, made it slower compared to when there were three performers and one was an expert. In an educational context, combining expert students with novice students seems sensible. This configuration would allow a teacher to correct and show to the learner(s) as necessary, to help students debug, and to suggest the beginnings of new computational or musical approaches that push the group beyond their boundaries. Turn-taking could also work for a non-hierarchical exchange between two persons with different skill sets.

Remote CMLC seems a suitable educational model in situations where the student cannot work from the classroom, e.g., when a student cannot commute, or when the student needs to work further an assignment. This approach can reinforce learning concepts beyond attending class and engage new students with Science, Technology, Engineering and Mathematics (STEM) fields in a longer term. This approach might also be useful for online courses.

5.2 CMLC and Turn-Taking within Performance

The experience of turn-taking recalls what Wessel and Wright [39] describe as the *catch and throw* metaphor in musical control, where there is a dialogue between a group of musicians, in which the musical material is received, modified, and sent in real time. In this case there is a shared space where the musical material is changed under request.

The model of turn-taking seemed a little bit slow for a performance setting, especially in remote collaboration. It would be interesting to combine it with Google Docs-like collaborative editing as discussed in [14]. It is still an open question how to combine the structure provided by turn-

taking with a more freestyle format provided by simultaneous multiple editors so that both performers and audience understand what is going on. Replacing the turn-taking hashtag mechanism with a GUI that mediates, shows, and tracks the state of users would improve turn-taking in performance. In remote collaboration, bringing a unified experience and a clear real-time representation of the different user states is expected from a suitable CMLC environment.

Collaborative error solving can be integrated in a performance setting by dividing roles (e.g., running the code privately before launching it to the shared space). Providing personal spaces in CMLC is in line with previous research on CSCW applied to music [26]. A balance needs to be found between debugging in individual spaces and paying attention to the driver's actions.

Algorithmic and random behaviors were explored in the code, and it is an open question whether certain behaviors could be configured from the DAW as part of the driver's role. In co-located collaboration, this would liberate the driver to the limited role of running the code. For example, switching between a manual and an automatic mode of running the code seems useful for both co-located and remote settings.

5.3 Online Chat, Semantic Hashtags, and Interaction Design

Using a chat window and hashtags was an effective tool of communication in the autoethnographic study. The hashtag language could include an easier way to get access to the information about the functions and parameters of the EarSketch API when trying to explain them in the chat. Supporting the use of own invented hashtags and vocabulary seems relevant here. Connecting hashtags to visible notifications that go to either a user or the group seems like the next step for improving the communication in EarSketch (see Section 4.2).

In remote collaboration, it was hard to tell if the others were active or not from the GUI. Making more visible the color-coding scheme for users, or showing active vs idle status, as preferred by the live coding community (see Section 4.2), should be addressed following design guidelines from CSCW literature. Adding an embedded chat next to the code is fundamental in both co-located and remote situations, as evidenced here.

6 CONCLUSION

This paper looked into co-located and remote turn-taking and online chatting in CMLC using the CS educational online platform EarSketch. We compared duo and trio live coding from an autoethnographic stance. We found that turn-taking in duo or trio live coding is more promising in education than in performance. We foresee that turn-taking and chatting in CMLC, between small groups of two, three, or four people, can be useful in the classroom for pedagogical purposes. We found that the role of a chat window is important as a tool for supporting communication in CMLC, but that our proposal of semantic hashtags should be reconsidered as a tailorable vocabulary adapted to the needs of each group and perhaps linked to a notification system that facilitates the collaboration. From our four use cases based on trio/duo vs co-located/remote situations, we discovered that a co-located trio live coding mediated by a turn-taking mechanism can be more interesting for group dynamics because the roles of a driver and two navigators can specialize and adapt easily during the musical improvisation act, while combining both verbal and non-verbal communication. However, one size does not fit all: we identified that a variety of group configurations (e.g., number of performers, types of roles) is preferred from the live coding community. We concluded that an hybrid form of turn-taking combined with multi-editing a shared script might be of more interest in performance. We plan to continue this research towards improving the collaboration support in EarSketch. We are interested in exploring real-world setups for both co-located and remote CMLC in education, which can contribute to CS and music education. Future work also includes gathering students response and exploring group dynamics.

7 ACKNOWLEDGMENTS

The authors are thankful to the participants of the study and to the reviewers of the manuscript for their suggestions. Most of the data collection and analysis of this research was carried out while the first author was at the Georgia Institute of Technology. The EarSketch project receives funding from the National Science Foundation (CNS #1138469, DRL #1417835, DUE #1504293, and DRL #1612644), the Scott Hudgens Family Foundation, the Arthur M. Blank Family Foundation, and the Google Inc. Fund of Tides Foundation.

8 REFERENCES

[1] S. Aaron, A. F. Blackwell, "From Sonic Pi to Overtone: Creative Musical Experiences with Domain-Specific and Functional Languages," presented at the *Proceedings of the First ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design*, pp. 35–46 (2013).

[2] J. Freeman, A. V. Troyer, "Collaborative Textual Improvisation in a Laptop Ensemble," *Computer Music J.*, vol. 35, no. 2, pp. 8–21 (2011).

[3] S. W. Lee, G. Essl, "Communication, Control, and State Sharing in Collaborative Live Coding," presented at the *Proceedings of the 14th International Conference on New Interfaces for Musical Expression*, pp. 263–268 (2014).

[4] C. Mckinney, "Quick Live Coding Collaboration in the Web Browser," presented at the *Proceedings of the 14th International Conference on New Interfaces for Musical Expression*, pp. 379–382 (2014).

[5] L. Williams, R. Kessler, *Pair Programming Illuminated* (Addison-Wesley Longman Publishing Co., Inc.) (2002).

[6] J. Freeman, B. Magerko, T. McKlin, M. Reilly, J. Permar, C. Summers, E. Fruchter, "Engaging Underrep-

resented Groups in High School Introductory Computing through Computational Remixing with EarSketch," presented at the *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pp. 85–90 (2014).

[7] G. Wakefield, C. Roberts, M. Wright, T. Wood, K. Yerkes, "Collaborative Live-Coding Virtual Worlds with an Immersive Instrument," *Proceedings of of the 14th International Conference on New Interfaces for Musical Expression* (2014).

[8] J. Rohrhuber, A. de Campo, *Just in Time Programming*, pp. 207–236 (The MIT Press) (2011).

[9] C. Roberts, K. Yerkes, D. Bazo, M. Wright, J. Kuchera-Morin, "Sharing Time and Code in a Browser-based Live Coding Environment," presented at the *Proceedings of the First International Conference on Live Coding*, pp. 179–185 (2015).

[10] T. Magnusson, "Confessions of a Live Coder," presented at the *Proceedings of the International Computer Music Conference 2011 (ICMC '11)*, pp. 609–616 (2011).

[11] A. Xambó, P. Shah, G. Roma, J. Freeman, B. Magerko, "Turn-Taking and Chatting in Collaborative Music Live Coding," presented at the *Proceedings of the 12th International Audio Mostly Conference on Augmented and Participatory Sound and Music Experiences*, pp. 24:1–24:5 (2017).

[12] Á. Barbosa, *Displaced Soundscapes: CSCW for Music Applications*, Ph.D. thesis, Universitat Pompeu Fabra (2006).

[13] A. McLean, "Reflections on Live Coding Collaboration," *Proceedings of the Third Conference on Computation, Communication, Aesthetics and X*, pp. 213–220 (2015).

[14] A. Xambó, J. Freeman, B. Magerko, P. Shah, "Challenges and New Directions for Collaborative Live Coding in the Classroom," presented at the *Proceedings of the International Conference of Live Interfaces* (2016).

[15] A. de Campo, "Republic: Collaborative Live Coding 2003–2013," presented at the A. Blackwell, A. McLean, J. Noble, J. Rohrhuber (Eds.), *Collaboration and Learning through Live Coding (Dagstuhl Seminar 13382)*, pp. 152–153 (2014 September).

[16] B. Taylor, "A History of the Audience as a Speaker Array," presented at the *Proceedings of the 17th International Conference on New Interfaces for Musical Expression*, pp. 481–486 (2017).

[17] G. Essl, "The Mobile Phone Ensemble as Classroom," presented at the *Proceedings of the 2010 International Computer Music Conference*, pp. 506–509 (2010).

[18] A. Ruthmann, J. M. Heines, G. R. Greher, P. Laidler, C. Saulters II, "Teaching Computational Thinking through Musical Live Coding in Scratch," presented at the *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, pp. 351–355 (2010).

[19] M. Edwards, "Algorithmic Composition: Computational Thinking in Music," *Communications of the ACM*, vol. 54, no. 7, pp. 58–67 (2011).

[20] A. R. Brown, "Editorial," *J. Music, Technology Educ.*, vol. 9, no. 1, pp. 3–4 (2016).

[21] J. Freeman, B. Magerko, "Iterative Composition, Coding and Pedagogy: A Case Study in Live Coding with EarSketch," *J. Music Teacher Educ.*, vol. 9, no. 1, pp. 37–54 (2016).

[22] R. M. Magee, C. M. Mascaro, G. Stahl, "Designing for Group Math Discourse," presented at the *Proceedings of the 2013 Conference on Computer Supported Collaborative Learning*, pp. 312–319 (2013).

[23] J. Fiala, M. Yee-King, M. Grierson, "Collaborative Coding Interfaces on the Web," presented at the *Proceedings of the International Conference of Live Interfaces*, pp. 49–58 (2016).

[24] C. Roberts, J. Kuchera-Morin, "Gibber: Live Coding Audio in the Browser," presented at the *Proceedings of the International Computer Music Conference 2012*, pp. 64–69 (2012).

[25] A. McLean, D. Griffiths, N. Collins, G. A. Wiggins, "Visualisation of Live Code," presented at the *Visualisation and the Arts*, pp. 1–5 (2010).

[26] R. Fencott, N. Bryan-Kinns, "Hey Man, You're Invading my Personal Space! Privacy and Awareness in Collaborative Music," presented at the *Proceedings of the 10th International Conference on New Interfaces for Musical Expression*, pp. 198–203 (2010).

[27] A. Xambó, G. Roma, P. Shah, J. Freeman, B. Magerko, "Computational Challenges of Co-creation in Collaborative Music Live Coding: An Outline," presented at the *Proceedings of the 2017 Co-Creation Workshop at the International Conference on Computational Creativity* (2017).

[28] H. Sacks, E. A. Schegloff, G. Jefferson, "A Simplest Systematics for the Organization of Turn-Taking for Conversation," *Language*, vol. 50, no. 4, pp. 696–735 (1974).

[29] A. McKinlay, R. Procter, O. Masting, R. Woodburn, J. Arnott, "Studies of Turn-Taking in Computermediated Communications," *Interacting with Computers*, vol. 6, no. 2, pp. 151–171 (1994).

[30] B. Jordan, A. Henderson, "Interaction Analysis: Foundations and Practice," *J. Learning Sciences*, vol. 4, no. 1, pp. 39–103 (1995).

[31] T. Blaine, S. Fels, "Contexts of Collaborative Musical Experiences," presented at the *Proceedings of the 3rd International Conference on New Interfaces for Musical Expression*, pp. 129–134 (2003).

[32] H. Anne-Marie, J. A. Hans, R. Pirkko, "Two Shared Rapid Turn Taking Sound Interfaces for Novices," presented at the *Proceedings of the 12th International Conference on New Interfaces for Musical Expression*, pp. 470–473 (2012).

[33] S. Fels, F. Vogt, "Tooka: Explorations of Two Person Instruments," presented at the *Proceedings of the 2002 Conference on New Interfaces for Musical Expression*, pp. 1–6 (2002).

[34] S. Aziz, M. Shamim, M. F. Aziz, P. Avais, "The Impact of Texting/SMS Language on Academic Writing of Students: What Do We Need to Panic About?" *Elixir Linguistics and Translation*, vol. 55, pp. 12884–12890 (2013).

[35] L. A. Shafie, N. Azida, N. Osman, "SMS Language and College Writing: The Languages of the College Texters," *International J. Emerging Technologies in Learning*, vol. 5, no. 1, pp. 26–31 (2010).

[36] R. Godwin-Jones, "Emerging Technologies: Messaging, Gaming, Peer-to-Peer Sharing: Language Learning Strategies & Tools for the Millennial Generation," *Language Learning & Technology*, vol. 9, no. 1, pp. 17–22 (2005).

[37] M. S. G. B. Hamzah, M. R. Ghorbani, S. K. B. Abdullah, "The Impact of Electronic Communication Technology on Written Language," *Online Submission*, vol. 6, no. 11, pp. 75–79 (2009).

[38] P. Dillenbourg, "What Do You Mean by 'Collaborative Learning'?" in P. Dillenbourg (Ed.), *Collaborative Learning: Cognitive and Computational Approaches*, vol. 1, pp. 1–19 (Elsevier, Oxford, UK) (1999).

[39] D. Wessel, M. Wright, "Problems and Prospects for Intimate Musical Control of Computers," *Computer Music J.*, vol. 26, no. 3, pp. 11–14 (2002).

APPENDIX I

Here are the five open questions that we used in our online questionnaire for the autoethnographic study:

1) *How was the experience of turn-taking live coding and chatting in a remote setting compared to your experience in a co-located setting?*
2) *How do you find the model of remote turn-taking for a performance setting?*
3) *Was the number of performers suitable for performance? Briefly explain why.*
4) *How do you find the model of remote turn-taking for an educational setting?*
5) *Was the number of performers suitable for an educational setting of remote turn-taking live coding? Briefly explain why.*

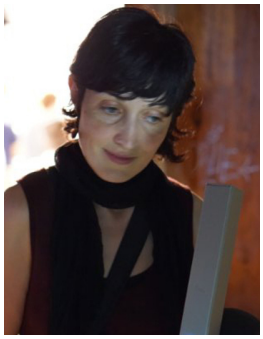
APPENDIX II

Here are the open questions and Likert item questions that we used in our online questionnaire for the live coding community study:

1) *What tools / technologies do you use for live coding? (e.g., SuperCollider, TidalCycles, Gibber, ixi lang, Sonic Pi, EarSketch, own-built environments/code...)*
2) *What tools / technologies do you use for collaborative music live coding? If different from above, briefly explain why.*
3) *What tools / technologies do you use / would you like to use for communicating between your peers in collaborative music live coding (e.g., embedded chat, video, instant messaging apps...)? Briefly explain why.*
4) *Do you prefer to work as either a duo live coding, a trio live coding or a group with more than three live coders? Briefly explain why. (Type N/A if not applicable).*
5) *What roles do the group members take? Do you switch roles? Briefly explain how and why. (Type N/A if not applicable).*
6) *In collaborative music live coding, I prefer to use emoticons to communicate between my peers during the performance or session. (1–5: Strongly disagree–Strongly agree, choose 0 if not applicable).*
7) *In collaborative music live coding, I prefer to use short messaging service (SMS) texting to communicate between my peers during the performance or session. (1–5: Strongly disagree–Strongly agree, choose 0 if not applicable).*
8) *In collaborative music live coding, I prefer to use provided hashtags (e.g. "#req", "#go", "#run") to communicate between my peers during the performance or session. (Strongly disagree–Strongly agree, choose 0 if not applicable).*
9) *In collaborative music live coding, I prefer to use our own invented hashtags to communicate between my peers during the performance or session. (Strongly disagree–Strongly agree, choose 0 if not applicable).*
10) *In collaborative music live coding, I prefer to have notifications on the screen from my other peers to communicate between us during the performance or session. (Strongly disagree–Strongly agree, choose 0 if not applicable).*
11) *In collaborative music live coding, I prefer to be able to comment on the side of the code during the performance or session. (Strongly disagree–Strongly agree, choose 0 if not applicable).*
12) *In collaborative music live coding, I prefer to use multimodal notifications (e.g., buzzers on the body, sounds) during the performance or session. (Strongly disagree–Strongly agree, choose 0 if not applicable).*
13) *In collaborative music live coding, I prefer to use a turn-taking mechanism mediated by the system during the performance or session. (Strongly disagree–Strongly agree, choose 0 if not applicable).*
14) *In collaborative music live coding, I prefer multi-editing a shared script (like in GoogleDocs) during the performance or session. (Strongly disagree–Strongly agree, choose 0 if not applicable).*
15) *In collaborative music live coding, I prefer that each live coder edits its own script during the performance or session. (Strongly disagree–Strongly agree, choose 0 if not applicable).*
16) *In collaborative music live coding, I prefer to use a color-coding scheme applied to represent the different peers who are contributing to the code/chat during the performance or session. (Strongly disagree–Strongly agree, choose 0 if not applicable).*
17) *In collaborative music live coding, I prefer to use a mechanism that shows active users vs. idle users during the performance or session. (Strongly disagree–Strongly agree, choose 0 if not applicable).*
18) *In collaborative music live coding, I prefer to use an embedded chat next to the code during the performance or session. (Strongly disagree–Strongly agree, choose 0 if not applicable).*
19) *In collaborative music live coding, what type of graphical user interfaces or chat interactions would you prefer?*

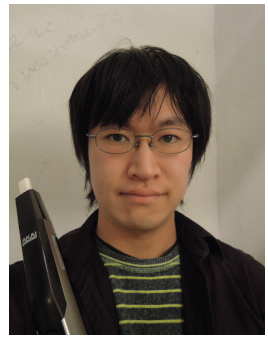
20) *If your system had a chat to communicate with the other group members, what kind of invented hashtags would you use? Briefly explain why.*
21) *How could the system shown in the video fit in your music creative workflow? Please, describe briefly.*
22) *How would you improve the system shown in the video to better improve your workflow?*
23) *Any additional comments or suggestions?*

THE AUTHORS


Anna Xambó


Gerard Roma


Pratik Shah


Takahiko Tsuchiya


Jason Freeman


Brian Magerko

Anna Xambó is a postdoctoral researcher at the Centre for Digital Music, Queen Mary University of London. She was previously a postdoctoral researcher at the Center for Music Technology and Digital Media Program (Georgia Tech). Her research looks into the design and evaluation of new interactive systems for music performance, publishing in international conferences and journals, such as NIME, ToCHI, CHI, TEI, and IwC. Her musical practice includes live coding, multichannel spatialization, tangible music, collaborative interfaces, audience participation with mobile devices, and real-time music information retrieval. Dr. Xambó is co-founder of the organization Women in Music Tech at Georgia Tech.

●

Gerard Roma received his Degree in Philosophy from Universitat Autònoma de Barcelona (UAB) in 1997. After several years working in software development, he obtained his M.Sc (2008) and PhD (2015) in Information and Communication Technologies from Universitat Pompeu Fabra (UPF). He is currently a Research Fellow at the Centre for Research in New Music (CeReNeM), University of Huddersfield (UK).

●

Pratik Shah is a graduate from Georgia Institute of Technology where he got his masters in Human-Computer Interaction. He worked on the EarSketch platform for 2 years during his graduate program. During this time he led all design efforts related to EarSketch. His interests lie in Interaction Design, Ubiquitous Computing and Story-telling and has published work in these areas before. He is currently working as an Experience Designer with the Global Print division at HP.

●

Takahiko Tsuchiya earned his master's degree in music technology from the Georgia Institute of Technology, where he continues as a doctoral student. His primary research interests are data sonification and analytics, including the development of data-agnostic frameworks and data encoding in musical structures. In the EarSketch project,

he develops the client-side technology including the public version of the real-time collaboration feature.

●

Jason Freeman is a Professor of Music at Georgia Tech. His artistic practice and scholarly research focus on using technology to engage diverse audiences in collaborative, experimental, and accessible musical experiences. He also develops educational interventions in K-12, university, and MOOC environments that broaden and increase engagement in STEM disciplines through authentic integrations of music and computing. His music has been performed at Carnegie Hall, exhibited at ACM SIGGRAPH, published by Universal Edition, broadcast on public radio's Performance Today, and commissioned through support from the National Endowment for the Arts. Freeman's wide-ranging work has attracted support from sources such as the National Science Foundation, Google, and Turbulence. He has published his research in leading conferences and journals such as Computer Music Journal, Organised Sound, NIME, and ACM SIGCSE. Freeman received his B.A. in music from Yale University and his M.A. and D.M.A. in composition from Columbia University.

●

Brian Magerko is Associate Professor of Digital Media in the School of Literature, Media, and Culture and a Human-Centered Computing faculty member in the School of Interactive Computing at Georgia Tech. He earned a B.S. in Cognitive Science from Carnegie Mellon University and a Ph.D. in Computer Science and Engineering from the University of Michigan. Dr. Magerko's work focuses on developing a better understanding of social collaboration and creativity between humans and artificial intelligence; designing and developing computational media experiences that inform and/or entertain; and using personal expression as a means of engaging the public–especially underrepresented populations–in computing.